Exploring Diffusion Models for Image Generation

Anu Agarwal[1], Mihir Pamnani[1], Pranav Sharma[1], and Shreya Shetye[1]

[1]University of Illinois Urbana-Champaign

May 11, 2024

1 Executive Summary

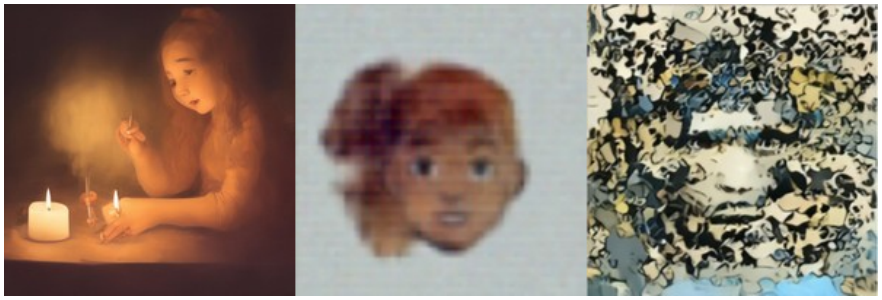


Fig. 1: Generated image samples from our three tasks (L-R) – Image for "a girl writing a letter in candlelight" by our fine-tuned Stable Diffusion Model ; Image imitating a cartoon generated by our Tiny-Diffusion in Pytorch ; Image of CelebA-Cartoon style generated by our Distilled Student Model on fine-tuned DDPM

Our project addresses limitations in generative modeling by fine-tuning Diffusion Probabilistic Models (DDPMs) [1] and Latent Diffusion Models (LDMs) [2] for generating cartoon or historic images and developing lightweight variants for local deployment. We overcome data coverage issues by fine-tuning DDPMs, enabling them to generate coherent images in previously unexplored domains. Additionally, we introduce a Tiny version of DDPM implemented in PyTorch [3] and utilize progressive distillation [4] to enhance efficiency and scalability, facilitating local deployment. Results demonstrate the effectiveness of our approach in generating high-quality images, even with limited data and computational resources. Our findings offer practical solutions for developing sophisticated generative models in resource-constrained environments, advancing the field of deep learning.

Key Components:

- Fine-tuning LDMs: Adapt established LDM architectures such as Stable Diffusion and Wuerstchen [5] for stylized text-to-image and masked inpainting.
- Tiny Version of DDPM in PyTorch: Develop a compact variant of DDPMs using the PyTorch framework, on the cartoon face dataset, which yields to generate better results compared to fine-tuning existing models.

- Tiny Version of DDPM with Progressive Distillation: Train compact student models with half the number of sampling steps from fine-tuned teacher models.

Interesting Findings:

- Fine-tuned LDMs, notably Stable Diffusion, effectively generated images matching the dataset style.
- Despite computational challenges, Stable Diffusion showed promise for in-painting on cartoon captions dataset.
- Extensive fine-tuning of DDPM on google-celeba-hq model produced exceptional outcomes on cartoon faces dataset.
- Training UNet on cartoon-faces dataset with diverse features. Completed within minutes on a single NVIDIA A100 GPU, demonstrating efficiency.
- Results from tiny-diffusion model suggest feasibility of training specialized UNet from scratch for specific tasks, providing viable alternatives to fine-tuning general-purpose models.
- Distilled CIFAR-10 images matched original DDPM quality with half the time steps, implying efficiency.
- Distilled student networks trained with source data and targets by Fine-tuned DDPM generate samples of visible face structure and cartoon-like characteristics with reduced fidelity.
- Distilled model provide 2x inference speed compared to fine-tuned DDPM, vital for resource-constrained environments.

2 Introduction

Diffusion models have gained remarkable traction in recent years for their ability to generate creative images and facilitate object replacement within images. Among these, Diffusion Probabilistic Models (DDPMs) have emerged as a promising avenue, distinguished by their capacity to produce high-quality images while accommodating diverse data distributions. These models operate by iteratively diffusing noise through multiple steps, gradually refining the generated images to achieve impressive sample quality.

In parallel, the exploration of Latent Diffusion Models (LDMs) has further advanced the capabilities of diffusion models. LDMs, including variants such as Stable Diffusion and Wuerstchen build upon the foundational principles of DDPMs while introducing enhancements to improve performance and efficiency. By leveraging latent variables, LDMs offer greater flexibility in modeling complex data distributions, enabling more nuanced and contextually relevant image generation.

Dreambooth [6], an innovative technique, offers a novel approach to fine-tuning pre-trained text-to-image diffusion models such as Stable Diffusion. Unlike traditional fine-tuning methods, Dreambooth associates specialized tokens and class prompts with input images during the fine-tuning process. This approach allows for the generation of concept-specific images by combining fine-tuned embeddings with text prompts. In contrast, LoRA [7], another fine-tuning method, directly modifies pre-trained weights during inference, enabling flexibility in generating images tailored to specific concepts or subjects.

Despite these advancements, challenges persist in effectively fine-tuning diffusion models for tasks such as stylized text-to-image and stylized masked inpainting. This project aims to address these challenges by leveraging techniques like Dreambooth and LoRA to enhance the capabilities of LDMs for generating visually compelling and semantically consistent image completions.

Through a multifaceted approach encompassing fine-tuning strategies, compact model implementations, and progressive distillation techniques, we seek to unlock the full potential of DDPMs and LDMs in addressing the intricate challenges posed by text-guided image generation and inpainting tasks. Ultimately, this project aims to not only advance the state-of-the-art in generative modeling but also enable practical applications in creative content generation, multimedia editing, and beyond.

2.1 Background

Diffusion models gradually add noise to an image during training, then remove it step by step during generation, inspired by thermodynamics principles. These models utilize fully convolutional networks like U-Net, which capture both high-level semantic information and low-level texture details. Instead of directly predicting denoised images, the model learns to predict the noise itself, scaled and subtracted during inference. Training involves inputting noisy images with varying noise levels and predicting added noise parametrized by the time step, enhancing stability and efficiency. The time step undergoes an embedding process for neural network compatibility. Compared to GANs [8], diffusion models offer greater stability as they progress through noise removal steps, reducing catastrophic failures.

DDPM scheduling orchestrates the gradual addition and removal of noise in diffusion probabilistic models during training and inference where it models image generation during the reverse process as a Markov Decision Process with the following formulation:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t,t), \Sigma_\theta(x_t,t)) \;\; \mu_\theta(x_t,t) \quad = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}t}}\epsilon\theta(x_t,t)\right) \;\; \Sigma_\theta(x_t,t) = \beta_t \mathbf{I} \tag{1}$$

It controls the rate and intensity of noise addition during training to optimize model convergence where the scheduling guides the step-by-step noise removal from noisy inputs to generate coherent images. Effective scheduling strikes a balance between exploration and exploitation, ensuring stable convergence and efficient exploration of the image space.

Following the popularity of DDPMs, Latent Diffusion Models (LDMs) were introduced in 2021 to improve the speed of image generation by limiting the diffusion process in a latent space. The architecture of LDMs utilizes several more components in addition to a DDPM/DDIM Scheduler including an Encoder, Decoder, U-Net, tokenizer, and a CLIP model. LDMs such as Stable Diffusion perform diffusion processes in latent space. It consists of a U-Net which is pre-trained to impose and remove noise from an image. An input text is provided to the CLIP model which uses semantic similarity between the text input and finds images similar to the input text using distance metrics such as 'Cosine distance' and 'Euclidean distance'. Once an image is associated with the text, it is provided as an input image to an encoder which compresses the image into a latent space. The diffusion process is performed on the compressed image by adding noise in the forward

process. During the backward pass, the added noise is removed by the U-Net and a clear image is provided to the decoder to reconstruct the image back to its original configuration. These models are widely used to perform a variety of tasks including text-to-image generation, Image Inpainting, Image Super-Resolution, and Image manipulation.

For fine-tuning diffusion models trained on large datasets, LoRA (Low-Rank Adaptation) was introduced which could modify the entire parameters of a model, enabling efficiency, and saving both time and computational resources. LoRA does it so by introducing two small weight matrices, which are used to update the weights of the infused matrices rather than updating the weights of the model directly.

$$W_0 + \Delta W = W_0 + AB \tag{2}$$

In the context of diffusion models, LoRA can be applied to the cross-attention layers within the U-Net that relate the image representations with the prompts that describe them. Because the original model is frozen and we inject new layers to be trained, we can save the weights for the new layers as a single file.

Lastly, in the context of our work, a significant downside of working with Diffusion models is often the high number of sampling steps. These steps represent the iterations needed during inference to generate images from isotropic Gaussian noise in the reverse process. In order to overcome the problem of high sampling steps, Progressive distillation was introduced by Salimans et al [4] which is an effective technique for transferring knowledge from large teacher models to compact student models. Firstly, it facilitates better knowledge transfer by gradually increasing the student model's capacity, allowing it to more effectively assimilate the teacher's knowledge compared to direct distillation to the final student size. Secondly, since they introduce a better distillation loss function through empirical findings, the progressive nature of the approach leads to more stable training and better convergence, mitigating the challenges often encountered when directly distilling to a large student model.

2.2 Our Approach

To achieve our objectives, we delineate our approach into three main components:

1. Fine-tuning LDMs: Adapting established LDM architectures like Stable Diffusion and Wrustchen for stylized text-to-image masked inpainting to improve the data coverage of these models by fine-tuning pre-trained models on historic and cartoon images.

2. Tiny Version of DDPM in PyTorch: Develop a compact variant of DDPMs by altering the U-Net and reverse process of diffusion.

3. Tiny Version of DDPM with Progressive Distillation: Leveraging knowledge from fine-tuned DDPMs, we perform progressive distillation to improve the speed of the diffusion process by training student models with fewer sampling steps.

Through rigorous empirical evaluations and qualitative analyses, we aim to showcase the efficacy and versatility of our approach in addressing the intricate challenges inherent in text-guided image generation and inpainting tasks. Ultimately, our endeavor aims to contribute to the development of robust and contextually aware generative models with broad applications across diverse domains.

3 Details of the approach

3.1 (Task 1) Fine-Tuning Diffusion and its applications

Firstly, our objective is to generate captions for a collection of historical images[1] utilizing BLIP [9]. Further, we will fine-tune a range of diffusion models, including SOTA Latent Diffusion Models (LDM), such as Stable Diffusion and Wuerstchen, on both the dlcvproj/historic_images[2] and Norod78/cartoon-blip-captions[3] datasets. Subsequently, we will perform in-painting using the fine-tuned Stable Diffusion model. Finally, we will fine-tune the google-celeb-hq DDPM on Skiittoo/cartoon-faces[4] dataset for later usage in Task 3.

DDPMs are U-Net [10] architectures that propagate forward and backward on an image. The forward pass involves adding noise to the clean image, while the backward pass includes removing the added noise to return a clean image. In contrast, LDMs include additional functionalities compared to DDPMs, such as the ability to generate images from text and perform in-painting on images. Both Stable Diffusion and Wuerstchen contain components such as Variational Auto Encoders (VAEs) [11], U-Net, and CLIP [12].

To fine-tune our LDMs for text-to-image fine-tuning, we use the Salesforce BLIP[5] image captioning model from HuggingFace to generate captions for our unlabeled dataset of historical images that do not consist of any captions. Captions are essential for this purpose. BLIP Fig.2 utilizes a Vision-Language Pre-training (VLP) framework that integrates understanding and generation tasks. This model is trained to generate a caption based on the content of an image and uses a patch embedding layer, a transformer encoder, and a language decoder. Once the images have captions, we push the dataset to the HuggingFace hub to be used for fine-tuning.

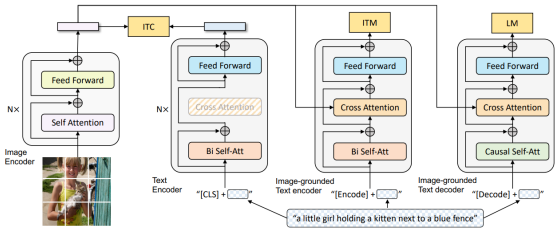


Fig. 2: BLIP architecture. [9]

After pushing the dataset to the Hugging Face hub, we used the historic images dataset and the cartoon captions dataset to fine-tune Stable Diffusion[6] and Wuerstchen[7] using the text-to-image diffusion files from the diffusers library.

After fine-tuning both the models for text-to-image tasks, we fine-tuned Stable Diffusion to perform

[1] https://huggingface.co/datasets/biglam/dating-historical-color-images
[2] https://huggingface.co/datasets/dlcvproj/historic_images
[3] https://huggingface.co/datasets/Norod78/cartoon-blip-captions
[4] https://huggingface.co/datasets/Skiittoo/cartoon-faces
[5] https://huggingface.co/Salesforce/blip-image-captioning-large
[6] https://huggingface.co/stabilityai/stable-diffusion-2-1
[7] https://huggingface.co/warp-ai/wuerstchen-prior

image in-painting using the cartoon captions dataset. We had to use a separate script from hugging face to perform fine-tuning for image in-painting using dreambooth.

After thorough refinement, we successfully fine-tuned the google-celebahq-256[8] DDPM from Hugging Face using the cartoon faces dataset. With 1000 sampling timesteps and a learning rate of 1e-5, we created a DDPM that is adept at generating cartoon-style faces. This technology can be extended to transform real-life user faces into cartoonized counterparts. In Task 3, we utilized the fine-tuned DDPM for distillation, significantly reducing the number of timesteps required for generating sample images.

3.2 (Task 2) Tiny Diffusion with Pytorch

In the contexts necessitating diffusion for narrow tasks, the adoption of conventional DDPMs poses significant challenges, notably in terms of time-intensive training procedures and computationally burdensome architectures. The conventional approach of training a full DDPM from scratch is inherently resource-intensive, rendering it unsuitable for deployment on edge devices due to their constrained computational capacities. To address these limitations, we aim to build and train a *tiny diffusion model* using PyTorch from scratch. After several experiments with different configurations for basic U-Net architecture for this tiny model, we have come up with the final architecture described in Fig. 3

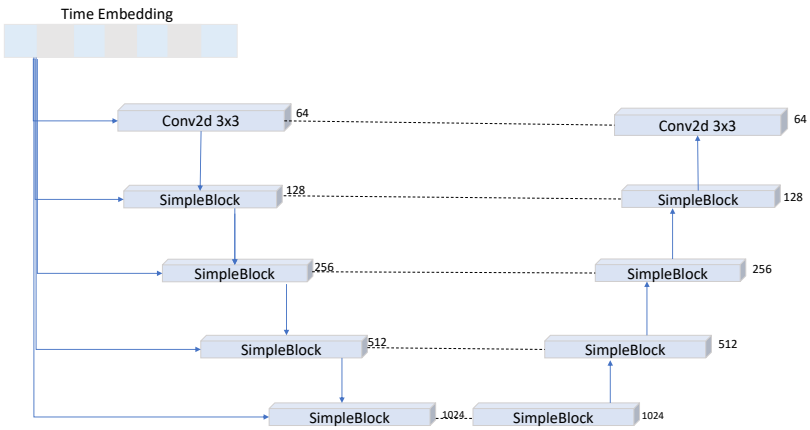


Fig. 3: U-Net Architecture for Tiny-Diffusion Model

The salient features which set our UNet architecture apart are:

- Interspersed Convolution and Self-Attention blocks in the SimpleBlock implementation
- Cosine Beta Scheduler
- SiLU activation instead of ReLU
- L2 loss

By compressing the U-Net architecture, we have reduced the number of trainable parameters from 113 million (in google/ddpmcelebahq_256 to 53 million in tiny diffusion. Details of the U-Net model architecture are described in Table-**??**.

[8] https://huggingface.co/google/ddpm-celebahq-256

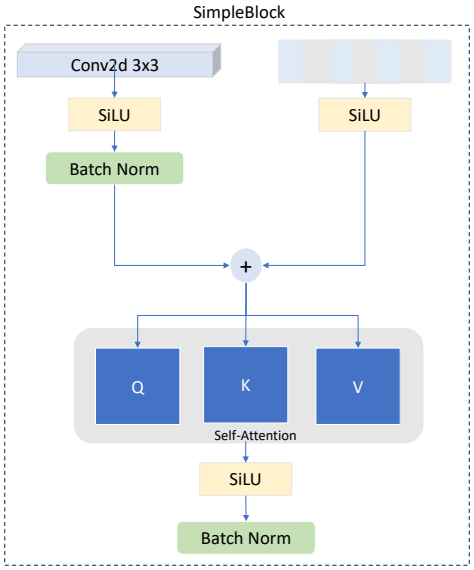


Fig. 4: Detailed Architecture of the SimpleBlock in U-Net

3.3 (Task 3) Tiny Diffusion with Progressive Distillation

With the aim of making the sampling process more manageable for instances with limited computing resources, the concept of Progressive Distillation was introduced by Salimans et al [4] as illustrated in Fig.5.

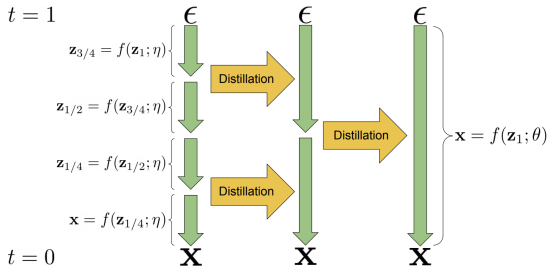


Fig. 5: Visualization of two iterations in Progressive Distillation

The authors of the paper exclusively employ a DDIM Sampler while training the student model using a target from the teacher network. They assert that adopting Progressive Distillation with a DDPM sampler would significantly compromise fidelity. However, their rationale for this claim remains limited, as all empirical experiments are conducted using a DDIM sampler. Furthermore, given the prevalent use of finely-tuned models, we propose to explore a novel approach: Distillation for Fine-tuned Diffusion models. To the best of our knowledge, existing literature lacks a clear framework for distilling fine-tuned diffusion models, supported by empirical evidence. For all our experiments, our source code stems from the Pytorch implementation [9] of Progressive Distillation

[9] https://github.com/Hramchenko/diffusion_distiller

inspired by the original TensorFlow implementation[10]. Thus, in this task, we perform Progressive Distillation for two kinds of diffusion models :

1. Sampling from a standard DDPM :

   In this sub-task, the target for the student network is the output generated by a diffusion model trained on the CIFAR-10 dataset. We obtained the model checkpoint for 1000 timesteps from `ema_diffusion_cifar10_model`[11]. Subsequently, we modified the U-Net architecture and network configurations, incorporating upsample and downsample blocks as defined by Ho et al. [1], to align the teacher model with the state dictionary of the pre-trained checkpoint. During the distillation process, the training images for the student network were resized to 32x32 dimensions, and we distilled the model for inference over 500 timesteps.

2. Sampling from a fine-tuned DDPM:

   In this task, the target for the student network is the output generated by our fine-tuned DDPM on the Skiittoo/cartoon-faces dataset. Given the lack of a standardized framework for distilling fine-tuned models, we conducted two experiments in this context: training the student model with the *source dataset* (the data used to train the diffusion model from scratch) and with the *fine-tuning dataset* (the data used to fine-tune the checkpoint). In both experiments of this sub-task, the network architecture resembled HuggingFace's implementation of *DDPMPipeline*[12] , which utilizes *UNet2DModel* and *DDPMScheduler*.

   To implement this approach, we obtained a pre-trained checkpoint with 1000 timesteps on the source dataset i.e. CelebA HQ, from google/ddpm-ema-celebahq-256[13]. To adapt this architecture for the teacher model, we adjusted the network configurations and imported the entire Diffusers[14] repository from HuggingFace into our project. For fine-tuning, we utilized the cartoon-faces dataset and obtained the checkpoint for the fine-tuned diffusion model from Task 1.

4 Results

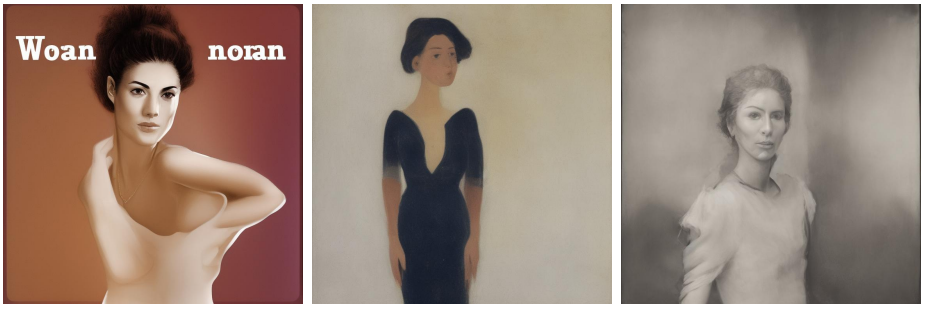
4.1 (Task 1) Fine-Tuning Diffusion and its applications

We employed the Salesforce BLIP5 image captioning model, accessible through Hugging Face, to generate captions for our historical image dataset lacking annotations. These captions were deemed crucial for fine-tuning our Latent Diffusion Models (LDMs) for text-to-image tasks. Upon obtaining captions, the dataset was uploaded to the Hugging Face hub as dlcvproj/historic_images[15] for subsequent fine-tuning processes.

Fine-tuning LDMs demonstrated impressive results for Stable Diffusion with the model able to generate images based on the style of the dataset they had been fine-tuned. The two fine-tuned models are dlcvproj/retro_sd_lora[16] and dlcvproj/cartoon_sd_lora[17].

[10] https://github.com/google-research/google-research/tree/master/diffusion_distillation
[11] https://github.com/pesser/pytorch_diffusion
[12] https://huggingface.co/docs/diffusers/v0.3.0/en/api/pipelines/ddpm
[13] https://huggingface.co/google/ddpm-ema-celebahq-256/tree/main
[14] https://github.com/huggingface/diffusers
[15] https://huggingface.co/datasets/dlcvproj/historic_images
[16] https://huggingface.co/dlcvproj/retro_sd_lora
[17] https://huggingface.co/dlcvproj/cartoon_sd_lora

(1a) (1b) (1c)



(2a) (2b) (2c)

(3a) (3b) (3c)

Fig. 6: Comparative analysis of images generated from (a) Stable Diffusions, (b) Cartoon caption fine-tuned Stable Diffusion, and (c) Historic images fine-tuned Stable Diffusion. Each row corresponds to a different prompt: (1) "a couple," (2) "a woman," and (3) "a girl writing a letter in candlelight."

Utilizing Stable Diffusion for in-painting on the cartoon captions dataset using our model dlcvproj/sd_inpainting_new[18] proved to be quite computationally intensive. We attempted to fine-tune the models over approximately 10000 iterations on the cartoon captions dataset. However, due to limited compute resources, we were only able to apply dreambooth fine-tuning for image in-painting for 2500 iterations on a g4dn.xlarge ec2 instance. The outcome was less than optimal, but we remain optimistic that better results could be achieved with additional compute resources in the future.

[18] https://huggingface.co/dlcvproj/sd_inpainting_new

(a)

(b)

Fig. 7: Comparison between results given by (a) Fine-Tuned Stable Diffusion on performing inpainting through prompt:'a <cmg >cartoon car', where <cmg >is the unique token representing the cartoon style and cartoon is the class prompt vs (b) Stable diffusion with prompt:'a car'.

We conducted extensive fine-tuning of DDPM on the pretrained google-celeba-hq model on the cartoon faces dataset, to generate a new DDPM dlcvproj/ddpm-celebahq-finetuned-cartoonfaces[19]. Our efforts yielded exceptional outcomes as shown in Fig.8, which we captured in the form of saved sampled images and a loss function plot as shown in Fig.9. Employing Mean Squared Error(MSE) loss, with a learning rate of 1e-5, we remain confident that this approach can be further developed to produce personalized cartoonized faces for any individual user.

[19] https://huggingface.co/dlcvproj/ddpm-celebahq-finetuned-cartoonfaces

(a)

(b)

Fig. 8: Comparison between sampled results generated by (a) google-celeba-hq and (b) google-celeba-finetuned-cartoonfaces fine-tuned on cartoon faces dataset

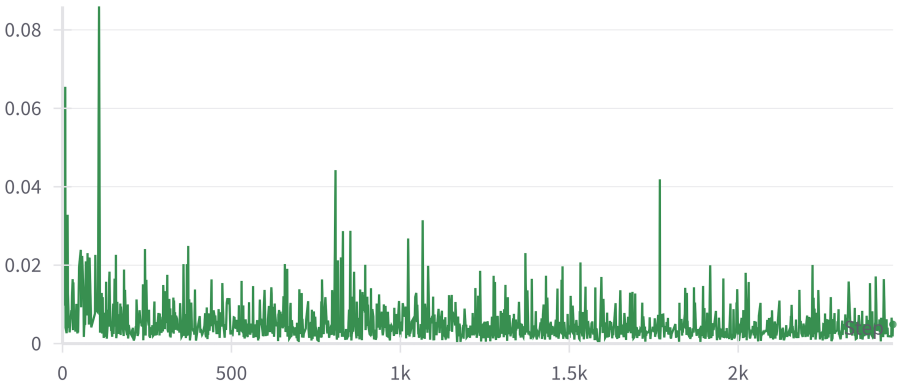
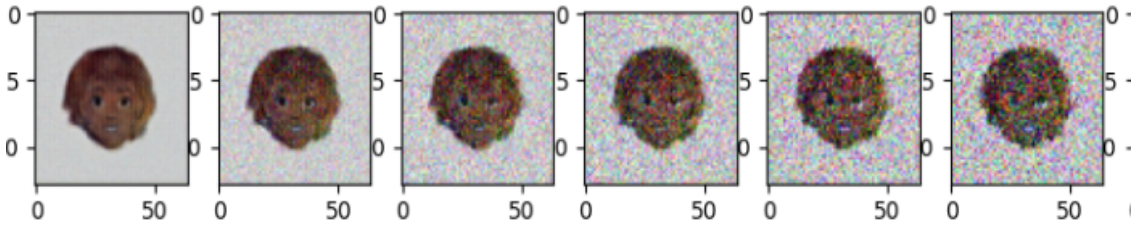
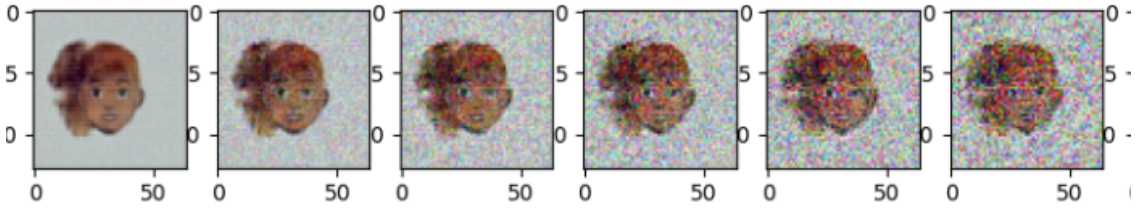


Fig. 9: Mean Square Error loss for fine-tuning google-celeba-hq for 2500 steps

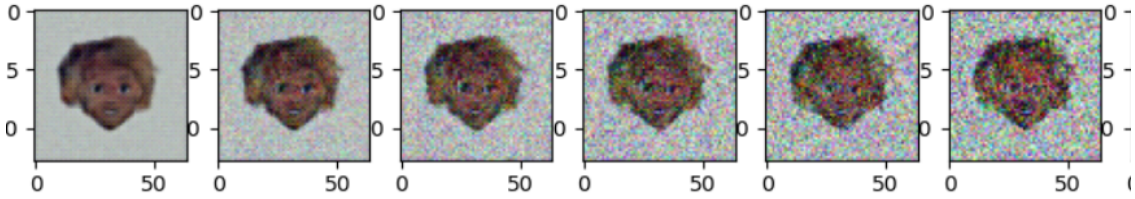
4.2 (Task 2) Tiny Diffusion with Pytorch



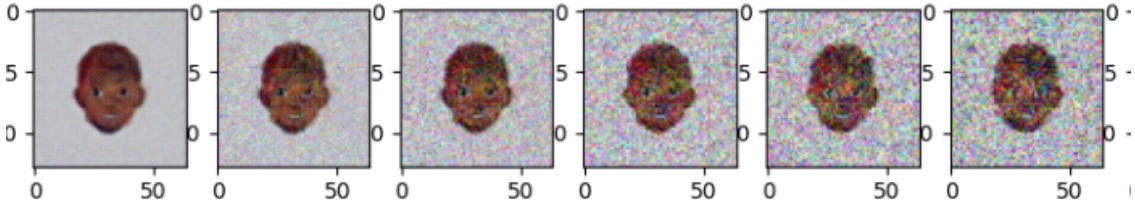
(a)



(b)



(c)



(d)

Fig. 10: Image generation results from Tiny-Diffusion

After training the UNet described in section 3.2 for nearly 60 epochs on 3000 training images from Skiittoo/cartoon-face dataset, we were able to obtain good quality cartoon faces (Fig.10) with good variation in facial features, hairstyles, hair color, complexion and expressions. The entire process took about 112 minutes to complete on one NVIDIA A100 GPU.

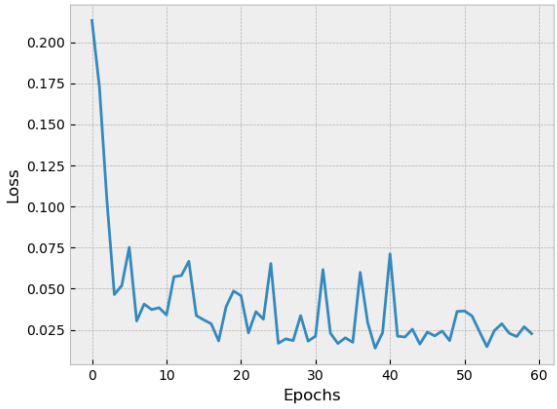


Fig. 11: Training Loss curve for Tiny-Diffusion

The loss curve for the training process is shown in Fig.11. The obtained results from tiny-diffusion model are sufficient to conclude that training a tiny UNet architecture from scratch for specific tasks with a narrow scope is an acceptable alternative to a fine-tuning a general purpose foundation model, which gives relatively low fidelity results (Fig.13).

The final set of hyperparameters used to obtain the results in Fig.10 are as follows:

- Image Resolution: 64x64
- Batch Size: 16
- Attention Heads: 8
- Learning Rate: 0.0001
- Learning rate decay: 0.95 after every 2 epochs
- n_timesteps: 1000
- Optimizer: Adam
- Epochs: 120

4.3 (Task 3) Tiny Diffusion with Progressive Distillation

The key motivation for this task was less sampling time and efficient learning of student network from teacher network in Progressive Distillation. Thus, the metrics of evaluation for this task are as follows:

1. Image Generation

   (a) Sampling from standard DDPM:

   In this sub-task, we worked with the progressive distillation of the CIFAR-10 dataset. As evident in Fig.12, the sample image generated by our distilled model is of similar visual quality as the original implementation with half the number of time steps.

   The hyper-parameters used for the above distillation are:

   - Image Resolution: 32x32
   - Batch Size: 1

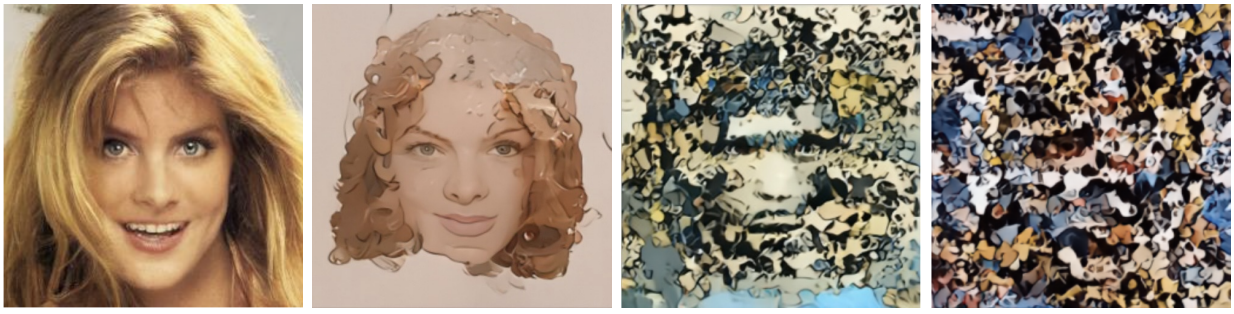
(a) (b)

Fig. 12: Comparison of images belonging to category='horse' in CIFAR-10 dataset generated by (a) Original Implementation with 1000 time steps (b) Our Distilled model with 500 timesteps

- Number of Images: 60,000
- n_iterations: 5000
- Learning Rate: 1.5e-05
- Scheduler: Linear
- Time Scale: 1.0
- n_timesteps (a): 1000
- n_timesteps (b): 500
- Optimizer: Adam

(b) Sampling from fine-tuned DDPM:

As discussed in the previous section, we have two experiments in this task i.e. training of student model with source dataset (CelebaHQ) and fine-tuning (cartoon) dataset. The samples of images generated by the distilled model with lesser time steps as compared to the original and fine-tuned DDPM are shown in Fig.13. We acknowledge that the fidelity of images generated by distilled models has reduced and the possible causes are discussed in the following section.



(a) (b) (c) (d)

Fig. 13: Comparison of random images generated by (a) Google's model trained on CelebaHQ dataset (b) Our Fine-tuned model trained on Cartoon dataset (c) Our Distilled model with student network trained on source dataset (CelebaHQ) (d) Our Distilled model with student network trained on fine-tuning dataset (cartoon-faces)

The hyper-parameters used for distillation from fine-tuned DDPM with source and fine-tuning dataset are similar to that of the previous sub-task with a few modifications:

- Image Resolution: 256x256
- Number of Images [(c) cartoon-faces dataset]: 5000
- Number of Images [(d) CelebaHQ dataset]: 3000
- n_timesteps (a, b): 1000
- n_timesteps (c, d): 500

2. Sampling Speed :

The inference speed was recorded by sampling images from Fine-tuned DDPM (1000 time steps) and Distilled DDPM (500 time steps) via HuggingFace API on Nvidia A100 GPU.

Number of Time Steps	**Inference Time**
1000	44 to 46 seconds
500	24 to 26 seconds

Table 1: Variation in inference time with change in number of time steps during the reverse process of diffusion.

The results show a nearly 2x speed-up in inference with the distilled model. With further steps of distillation, the speed is only expected to improve while keeping the fidelity of generated images intact. It must be noted that this is a significant increase in sampling when considering local CPU compute resources where it takes around 60 minutes for 100 sampling steps.

3. Distillation loss

The loss function used for Progressive Distillation is the standard reconstruction loss (mean squared error) defined by predicting v as mentioned by Salimans et al [4] :

$$\ell_\theta = \|\mathbf{v}_t - \hat{\mathbf{v}}_t\|_2^2 = \left(1 + \frac{\alpha_t^2}{\sigma_t^2}\right) \|\mathbf{x} - \hat{\mathbf{x}}_t\|_2^2$$

where : $\mathrm{v} \equiv \alpha_t \boldsymbol{\epsilon} - \sigma_t \mathbf{x}$, which gives $\hat{\mathbf{x}} = \alpha_t \mathbf{z}_t - \sigma_t \boldsymbol{v}_0(\mathbf{z}_t)$.

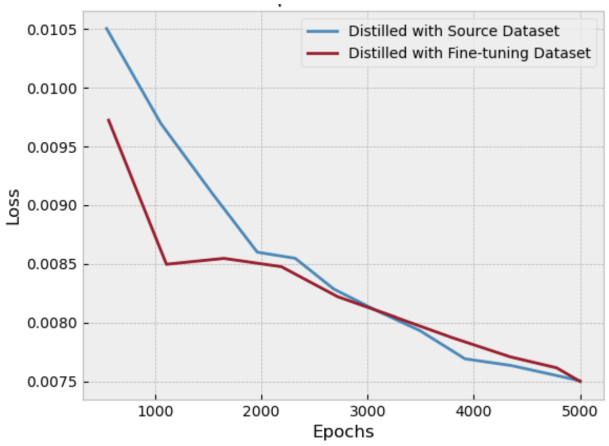
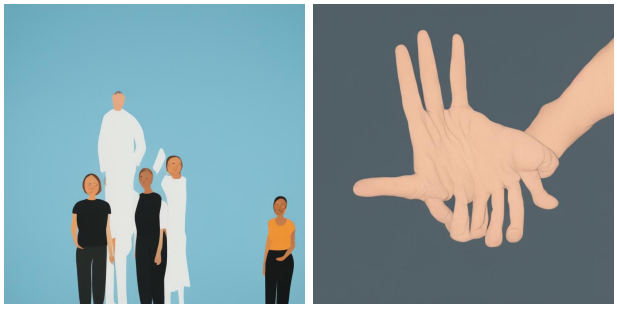


Fig. 14: Loss curves while sampling from fine-tuned DDPM, when the student network learns from the teacher network using the source dataset (CelebaHQ) and fine-tuning dataset (cartoon).

5 Discussion and conclusions

(**Task 1**) During the fine-tuning of LDMs, we explored various adjustments to enhance performance:



(a) (b)

Fig. 15: Artifacts observed in fine-tuned LDM

- Our model struggles in explicitly understanding count. When given a prompt to generate 'Two individuals' Fig.15(a), it generates an image of multiple individuals, and further, our model like other diffusion models struggles in generating complicated images such as human hands Fig.15(b).
- Reducing the batch size to 1 and resizing images to 128 significantly improved image inpainting tasks, facilitating more frequent updates to model parameters and accommodating resource constraints, respectively.
- Addressing an error related to inpainting channel size in diffusion model development involved implementing a custom compressor class within the diffusers, ensuring compatibility with the model architecture.
- Increasing the number of timesteps in DDPMs notably boosted model performance, allowing for more iterations to refine generated images.
- Achieving optimal training required fine-tuning the learning rate to 1e-5, underscoring the importance of hyperparameter tuning.
- Switching from the Norod78/cartoon-blip-captions dataset to the Skiittoo/cartoon-faces dataset simplified the process and improved performance by exclusively containing cartoon faces, easing translation into the celebrity faces dataset.
- Despite efforts to fine-tune Wrustchen and Kandinsky using an AWS EC2 instance, persistent errors related to resource limitations hindered proper fine-tuning.

(**Task 2**) While experimenting with different model architectures and hyperparameters for tiny-diffusion model, we observe the following the following interesting trends:

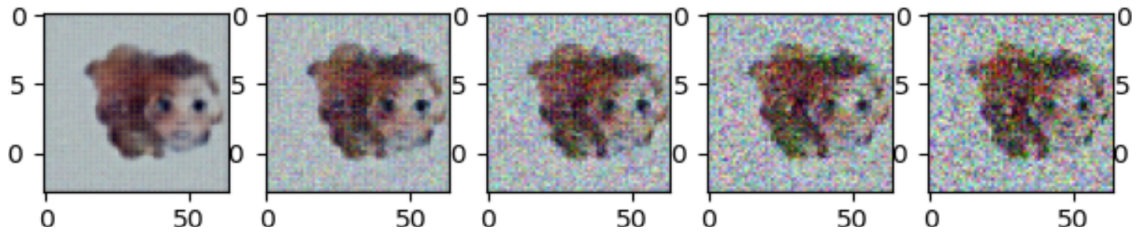


Fig. 16: Artifacts observed in Tiny-Diffusion samples

- Replacing Attention block inside the SimpleBlock with Convolutional block significantly deteriorates the model performance. We do not see any discernible cartoon faces in the outputs even after 50 training epochs.
- Increasing the number of attention heads tremendously improved model performance. With 8 attention heads, face outlines started emerging after epoch 2 itself while they did not emerge until epoch 8 with 2 heads.
- Upon experimenting with different no. of training samples, we observe that larger training set not only leads to more variation in the generates samples, but also makes the model converge faster.
- Tiny diffusion model, like all other diffusion models we experimented with, as a part of this project, experiences difficulty in the count of main objects to be generated in the image. We observed that certain images generated by tiny-diffusion had two faces instead of one (Fig.16).

(**Task 3**) With progressive distillation of a standard DDPM sampler, it is noticed that there is hardly any loss in fidelity of image generation with half the number of time steps. Whereas when we try to distill a fine-tuned DDPM, we make some surprising observations :

- When the student network is trained with the fine-tuning dataset and target as output from fine-tuned DDPM, the quality of generated images is considerably poor lacking any facial structure as shown in Fig.13 (d)
- When the student network is trained with the source dataset and target as output from fine-tuned DDPM, the quality of generated images is better with visual structure and significant cartoon-like characteristics visible as shown in Fig.13 (c)

Based on empirical evidence, we can assert the following regarding the framework for distilling fine-tuned models:

The distilled student networks can autonomously learn fine-tuned features from the teacher network's weights. However, when it comes to features from the source dataset, they benefit immensely from incorporating the source dataset during training to learn low-level features and patterns directly from the data, which may not be fully encoded in the teacher's weights.

We acknowledge that the visual quality of the output images generated by our distilled models on fine-tuned DDPM is not high. This can be attributed to two key factors: the limited capacity of the fine-tuned teacher model and the relatively small number of source dataset images used during the distillation process. Given the adequate compute resources overcoming the mentioned limitations, the visual quality of generated images is expected to improve significantly.

Our findings underscore the importance of striking a careful balance between learning from the

teacher model's knowledge and learning directly from the source data during the knowledge distillation process.

6 Statement of individual contribution

- Pranav Sharma: Pranav has performed fine-tuning of Stable Diffusion using dreambooth to perform image inpainting on the cartoon-caption dataset, as well as fine-tuning DDPM on the cartoon-faces dataset, which had been later used to perform Progressive Distillation. He further carried out the fine-tuning of Wuerstchen on the cartoon-caption dataset and has uploaded all these models to our Hugging Face hub account.
- Shreya Shetye: Shreya made use of BLIP to generate captions for the historic images dataset and uploaded the final model on Hugging Face. She further fine-tuned the Stable Diffusion Models on the cartoon dataset and the historic images dataset. She also carried out the fine-tuning of Wuerstchen and Kandinsky models on the cartoon dataset.
- Anu Agarwal: Anu performed extensive experimentation with different activations, usage of attenvion v/s convolutional blocks in U-Net, and various scheduler strategies to arrive at the final model architecture for the tiny-diffusion model in Pytorch. She also helped Mihir with the data pipeline setup for progressive distillation, along with the integration of fine-tuned DDPM models with the distillation pipeline.
- Mihir Pamnani: Mihir primarily worked with Progressive Distillation of the DDPM Sampler and Fine-tuned DDPM Sampler by analyzing network configurations and extensive experimentation with the balance of data and learning from teacher weights. He also helped Anu with the data pipeline setup and strategies for model architecture in the Pytorch implementation of Tiny Diffusion for optimal image generation.

References

[1] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[2] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models. 2022 ieee," in *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10674–10685, 2021.

[3] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[4] T. Salimans and J. Ho, "Progressive distillation for fast sampling of diffusion models," *arXiv preprint arXiv:2202.00512*, 2022.

[5] P. Pernias, D. Rampas, M. L. Richter, C. Pal, and M. Aubreville, "Würstchen: An efficient architecture for large-scale text-to-image diffusion models," in *The Twelfth International Conference on Learning Representations*, 2023.

[6] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman, "Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. 2023 ieee," in *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22500–22510, 2022.

[7] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[9] J. Li, D. Li, C. Xiong, and S. Hoi, "Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation," in *International conference on machine learning*, pp. 12888–12900, PMLR, 2022.

[10] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pp. 234–241, Springer, 2015.

[11] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[12] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, pp. 8748–8763, PMLR, 2021.